

# **Mobile FBUS 1.5**

## **Programming Reference**

Copyright © 2000 by Gert-Jaap Glasbergen, Berend Engelbrecht. All rights reserved.

Mobile FBUS is a Software Cave product  
<http://www.softwarecave.com>  
[info@softwarecave.com](mailto:info@softwarecave.com)

## Table of Contents

1	About Mobile FBUS 1.5 .....	3	4	Events .....	9
2	Properties .....	4	4.1	IncomingCall event.....	9
2.1	ACDCPower property.....	4	4.2	LogoAnimation event.....	9
2.2	AlarmEnabled property.....	4	5	Collections .....	10
2.3	AlarmTime property.....	4	5.1	NetmonitorScreen .....	10
2.4	AnimateLogo property.....	4	5.2	PhoneBook.....	10
2.5	BatteryLevel property.....	4	5.3	SpeedDial .....	11
2.6	CallInProgress property.....	4	6	SMS Object.....	12
2.7	CellId property.....	4	6.1	Properties .....	12
2.8	Connected property.....	4	6.1.1	SMS.LastError property.....	12
2.9	CurrentCallNr property.....	4	6.1.2	SMS.TotalMessages property.....	12
2.10	DateTime property.....	5	6.1.3	SMS.UnreadMessages property.....	12
2.11	IncomingCallInProgress property.....	5	6.2	Methods .....	12
2.12	KeyboardLocked property.....	5	6.2.1	SMS.Refresh method .....	12
2.13	LAC property.....	5	6.2.2	SMS.SendMessage method .....	13
2.14	ProviderCode property.....	5	6.3	Sub-objects .....	13
2.15	ProviderCountry property.....	5	6.3.1	SMS.DeliveryNotifications/Inbox/Outbox objects .....	13
2.16	ProviderName property.....	5	6.3.2	FBSmsMessage object .....	13
2.17	ProviderSMSC property.....	5	7	Logo object .....	14
2.18	RfLevel property.....	5	7.1	Properties .....	14
2.19	SMSCClass property.....	5	7.1.1	Logo.Height property.....	14
2.20	SMSSettingSet property .....	6	7.1.2	Logo.LastError property.....	14
3	Methods.....	7	7.1.3	Logo.LogoType property.....	14
3.1	About method.....	7	7.1.4	Logo.NetCode property .....	14
3.2	Connect method .....	7	7.1.5	Logo.Pixel property .....	14
3.3	DisableNetMonitor method .....	7	7.1.6	Logo.Width property.....	15
3.4	Disconnect method .....	7	7.2	Methods .....	15
3.5	EnableNetMonitorEngineering method.....	7	7.2.1	Logo.Download method.....	15
3.6	EnableNetMonitorFieldTest method.....	7	7.2.2	Logo.ReadFile method .....	15
3.7	LoadAnimation method .....	8	7.2.3	Logo.SendAsSMS method .....	15
3.8	SendDtmf method .....	8	7.2.4	Logo.Upload method .....	15
3.9	SetLogoPicture method .....	8	7.2.5	Logo.WriteFile method.....	16
3.10	SetStartupText method.....	8			
3.11	VoiceCall method .....	8			

## **1 About Mobile FBUS 1.5**

This document describes the use of the Mobile FBUS ActiveX Control in Visual Basic 6.0.

Mobile FBUS 1.5 is a freeware ActiveX control that you can use to create software for mobile phones. Send SMS messages, manipulate operator logos, use monitoring, access phonebook, and much, much more. Mobile FBUS is the ideal tool for connecting your Visual Basic application to mobile phones.

Mobile FBUS works with Nokia GSM phones, we test our code with Nokia 51xx/61xx and 3210 models.

## 2 Properties

*Note: most properties of the Mobile FBUS ActiveX control are only available when the control is connected to a telephone.*

### 2.1 ACDCPower property

*Boolean - Read only*

This property returns True when the phone is connected to an external power source, e.g., a battery loader.

### 2.2 AlarmEnabled property

*Boolean - Read only*

This property returns True when the alarm clock of the phone is enabled.

### 2.3 AlarmTime property

*Date - Read/Write*

This property returns the time to which the alarm clock of the connected phone is set. If you assign a time, the alarm clock is set to that time and is enabled.

Note: You cannot set the seconds; the date and seconds parts of the VB Date are ignored when you set the alarm time..

### 2.4 AnimateLogo property

*Boolean - Read /Write*

This property can be used to start (set to True) or stop (set to False) logo animation. You must load a logo animation using the LoadAnimation method before AnimateLogo has any effect.

The operator logo is only animated as long as the FBUS cable is connected and your software is running.

### 2.5 BatteryLevel property

*Single - Read only*

This property returns the battery level of the phone. The units that are used depend on the phone model.

### 2.6 CallInProgress property

*Boolean - Read only*

This property returns True if a call (incoming or outgoing) is in progress on the connected phone. Check the IncomingCallInProgress property to find out if the call is incoming or outgoing.

### 2.7 CellId property

*String - Read only*

This property returns the *Cell ID* of the network node to which the phone is currently connected. The Cell Id is unique to each transmitter of the GSM network, whereas the Location Area Code (LAC) is common between a group of transmitters in the same area.

### 2.8 Connected property

*Boolean - Read only*

This property returns if the control has made a link to the phone.

### 2.9 CurrentCallNr property

*String - Read only*

This property returns the phone number of the remote party if a call is in progress.

## 2.10 DateTime property

*Date - Read/Write*

This property returns or sets the current date and time of the internal clock in the connected phone.

Note:

1. You cannot set the seconds; the seconds part of the VB Date is ignored when you set the date/time.
2. In a Nokia 51xx series phone you cannot set the date through the user interface of the phone, so it may contain a random value when you query it. However, you can set the date through Mobile FBUS and your phone will keep the correct date from then on.

## 2.11 IncomingCallInProgress property

*Boolean - Read only*

This property returns True if an incoming call is in progress on the connected phone.

## 2.12 KeyboardLocked property

*Boolean - Read only*

This property returns True if the keyboard of the connected phone is disabled.

## 2.13 LAC property

*String - Read only*

This property returns the *Location Area Code* of the network node to which the phone is connected. The Cell Id is unique to each transmitter of the GSM network, whereas the Location Area Code is common between a group of transmitters in the same area.

## 2.14 ProviderCode property

*String - Read only*

This property returns the provider code for the current network provider in the format "ccc nn", where ccc is the three-digit country code and nn is the network code. For example, the dutch provider Ben has provider code "204 16".

## 2.15 ProviderCountry property

*String - Read only*

This property returns the country of the cellular provider the phone is currently connected to.

## 2.16 ProviderName property

*String - Read only*

This property returns the name of the cellular provider the phone is currently connected to.

## 2.17 ProviderSMSC property

*String - Read only*

This property returns the phone number of the Short Message Service Center (SMSC) of the cellular provider the phone is currently connected to.

## 2.18 RfLevel property

*Single - Read only*

This property returns the GSM signal strength as detected by the phone. The units that are used depend on the phone model.

## 2.19 SMSCClass property

*Integer - Read/Write*

This property returns or sets the SMS message class that will be used in the SendSMSMessage method. The SMS message class defines how the message is displayed by the receiving phone. The following values may be used:

fbSmsClassFlash	Message is shown as flash message (immediately show in main display of the telephone without user confirmation)
fbSmsClassNormal	Message is handled as normal SMS message.
fbSmsClass2	Same effect as fbSmsClassNormal in Nokia telephones.

fbSmsClass3      Same effect as fbSmsClassNormal in Nokia telephones.

## **2.20 SMSSettingSet property**

*String - Read/Write*

This property returns or sets the used set of SMS settings. This is the used set of settings containing the SMS Centre, type of SMS etc. On a Nokia phone there are always minimum three sets. Default value is 1.

## 3 Methods

### 3.1 About method

*Sub About*

This method opens a window that shows copyright and version information of the ActiveX control.

**Parameters:**

None

**Returns:**

Nothing

### 3.2 Connect method

*Sub Connect [sComport As String = "COM1"]*

This method is used to connect the control to the mobile phone. This is essential to do, as no other method will work if the Connect method has not been called.

**Parameters:**

sComport -- Optional -- The port to connect to. If left out, COM1 will be used. Enter the name of the comport like "COM<number>" (e.g. COM1, COM2, COM3 etc.)

**Returns:**

Nothing

### 3.3 DisableNetMonitor method

*Sub DisableNetMonitor*

This method will disable the Netmonitor menu on the connected phone.

**Parameters:**

None

**Returns:**

Nothing

### 3.4 Disconnect method

*Sub Disconnect*

This method will end the connection to the phone, and with it it will disable all other methods in the control.

**Parameters:**

None

**Returns:**

Nothing

### 3.5 EnableNetMonitorEngineering method

*Sub EnableNetMonitorEngineering*

This method will enable the engineering netmonitor on the connected phone.

**Parameters:**

None

**Returns:**

Nothing

### 3.6 EnableNetMonitorFieldTest method

*Sub EnableNetMonitorFieldTest*

This method will enable the field test netmonitor on the connected phone.

**Parameters:**

None

**Returns:**

Nothing

### 3.7 LoadAnimation method

*Sub LoadAnimation(sGifFile As String, [iBrightnessThreshold As Integer = 128])*

This method reads an animated gif file to be used in logo animation. After loading the animation you must set the `AnimateLogo` property to `True` to start the animation.

Note:

1. The operator logo is only animated as long as the FBUS cable is connected and your software is running.
2. It is recommended to use animated bitmaps of exactly 144 x 28 pixels. Large or very small animated gifs may not be rendered correctly (or not at all).

**Parameters:**

`sGifFile` -- Mandatory -- The animation to be used

`iBrightnessThreshold` -- Optional -- Pixels that have less luminance (brightness) than this level will be shown as black on the phone display, above this level they will be shown as white. The valid range for this parameter is between 0 and 255.

**Returns:**

Nothing

### 3.8 SendDtmf method

*Sub SendDtmf(sNumber As String)*

This method will send a sequence of DTMF tones to the phone. This can be used to control voice response systems that respond to tone-dialing signals. You must have a voice call in progress for this to work.

**Parameters:**

`sNumber` -- Mandatory -- String containing digits to be sent as tones.

**Returns:**

Nothing

### 3.9 SetLogoPicture method

*Sub SetLogoPicture(Picture As IPictureDisp, [iBrightnessThreshold As Integer = 128], [IScreenColor = &HFF00FF])*

This method translates a Visual Basic picture property to a logo picture. When successful, the logo picture will be available in the `Logo` object after `SetLogoPicture` returns.

**Parameters:**

`Picture` -- Mandatory -- Picture reference. You must pass the `Picture` property of a `PictureBox` or `Image` control in this parameter.

`iBrightnessThreshold` -- Optional -- Pixels that have less luminance (brightness) than this level will be shown as black on the phone display, above this level they will be shown as white. The valid range for this parameter is between 0 and 255.

`IScreenColor` -- Optional -- This color is used as transparency mask color. By default saturated magenta (R,G,B=255,0,255) is used for this.

**Returns:**

Nothing

### 3.10 SetStartupText method

*Sub SetStartupText(sText As String)*

This method will set the text `sText` as startup text.

**Parameters:**

`sText` -- Mandatory -- String containing the new startup text

**Returns:**

Nothing

### 3.11 VoiceCall method

*Sub VoiceCall(sNumber As String)*

This method will initiate a voice call to the number specified in the `sNumber` parameter.

**Parameters:**

`sNumber` -- Mandatory -- String containing the phone number to be called

**Returns:**

Nothing



## 4 Events

### 4.1 IncomingCall event

*Event IncomingCall(sIncomingNumber As String)*

This event occurs when a call is coming in while the control is connected to the phone. The event passes the phone number of the caller in sIncomingNumber.

**Parameters:**

sIncomingNumber -- A string containing the phone number of the incoming call.

### 4.2 LogoAnimation event

*Event LogoAnimation(iFrame As Integer, iTotFrames As Integer)*

This event occurs each time after a frame has been sent to the phone in an operator logo animation.

**Parameters:**

iFrame -- The frame number of the frame that just has been uploaded. Frame numbers range from 0 to iTotFrames-1.

iTotFrames -- Total number of frames in the animation.

## 5 Collections

### 5.1 NetmonitorScreen

*NetmonitorScreen(nMode As Integer) As String*

This collection can be used to query the different net monitor screens when the engineering or field test mode is enabled. The contents of each monitor screen is returned in a string. The net monitor does not have to be visible on the phone display, but it must be enabled for the NetmonitorScreen collection to be valid.

**Parameters:**

nMode -- Mandatory -- Screen number to be returned

### 5.2 PhoneBook

*PhoneBook(iLocation As Integer, [iMemoryType As fbMemoryType = fbSimMemory]) As FBPhonebookEntry*

This collection can be used to manage the Phonebook stored in your SIM or phone memory.

**Parameters:**

iLocation -- Mandatory -- Memory location to be read or written.

iMemoryType -- Optional -- Type of memory to be accessed. If this parameter is left out, the SIM memory will be used.

**Properties:**

Group -- Read/Write Integer -- The Group ID for this phonebook entry, 0 if the entry is not part of a group.

LastError -- Read only Integer -- Last error from a phonebook action. 0 if the action was successful.

Name -- Read/Write String -- The name for this phonebook entry

Number -- Read/Write String -- The telephone number for this phonebook entry

**Methods:**

WriteData -- Use this to write Name, Number and Group ID in one action. This is faster than assigning the three properties one by one.

**Usage Example:**

The following sample code reads all phonebook entries from the SIM and prints the data to the VB debug monitor.

```
MobileFBUSControl1.Connect
For i = 1 To 255
    If MobileFBUSControl1.PhoneBook(i).Number <> "" Then
        Debug.Print MobileFBUSControl1.PhoneBook(i).Name, _
                    MobileFBUSControl1.PhoneBook(i).Number, _
                    MobileFBUSControl1.PhoneBook(i).Group
    ElseIf MobileFBUSControl1.PhoneBook(i).LastError = 22 Then
        Debug.Print "End of SIM memory, (max. " & CStr(i - 1) & " entries)"
        Exit For ' End of SIM reached
    End If
Next
MobileFBUSControl1.Disconnect
```

### 5.3 SpeedDial

*SpeedDial(iLocation As Integer) As FBSpeedDial*

This collection manages the Speed Dial entries in your phone's memory. A speed dial in a Nokia phone is a *reference* to a phone book entry in either SIM or telephone memory.

**Parameters:**

iLocation -- Mandatory -- Speed dial location to be read or written.

**Properties:**

LastError -- Read only Integer -- Last error from a speed dial action. 0 if the action was successful.

MemoryType -- Read/Write Integer -- The memory type for this speed dial entry. Can be either fbPhoneMemory (0) or fbSimMemory (1).

Number -- Read/Write Integer -- The phonebook location for this speed dial entry. 0 if the speed dial is not in use.

**Usage Example:**

The following sample code reads the first speed dial entry and prints the data of the speed dial and the associated phonebook entry to the VB debug monitor.

```
Dim iNumber as Integer
Dim iMemoryType as fbMemoryType

MobileFBUSControl1.Connect
iNumber = MobileFBUSControl1.SpeedDial(1).Number
iMemoryType = MobileFBUSControl1.SpeedDial(1).MemoryType
Debug.Print "SpeedDial 1:", iNumber, iMemoryType
Debug.Print "Name:", MobileFBUSControl1.PhoneBook(iNumber,iMemoryType).Name
Debug.Print "Number:", MobileFBUSControl1.PhoneBook(iNumber,iMemoryType).Number
MobileFBUSControl1.Disconnect
```

## 6 SMS Object

This object can be used to read and send SMS messages. The SMS object has the following properties, methods and sub-objects:

Property	Description
LastError	Returns the last error code.
TotalMessages	Returns the total number of SMS messages stored.
UnreadMessages	Returns the number of unread messages.
Method	Description
Refresh	Retrieves stored messages from the phone.
SendMessage	Sends an SMS message.
Object	Description
DeliveryNotifications	Message box containing delivery notifications.
Inbox	Message box containing incoming messages.
Outbox	Message box containing sent messages.
FBSmsMessage	Sub-object representing one message in a message box.

### 6.1 Properties

#### 6.1.1 SMS.LastError property

*Integer - Read only*

This property returns the last internal error that occurred when calling an SMS property or method. LastError is 0 when the last operation completed normally.

#### 6.1.2 SMS.TotalMessages property

*Integer - Read only*

This property returns the total number of SMS messages stored in SIM memory. This property returns the current value for the connected phone, even if the SMS.Refresh method has not been called.

#### 6.1.3 SMS.UnreadMessages property

*Integer - Read only*

This property returns the number of unread SMS messages stored in SIM memory. This property returns the current value for the connected phone, even if the SMS.Refresh method has not been called.

### 6.2 Methods

#### 6.2.1 SMS.Refresh method

*Sub Refresh()*

This method refreshes the lists of messages stored in the SMS.Inbox, SMS.Outbox and SMS.DeliveryNotification sub-objects. SMS.Refresh must be called at least once if you want to use the message box objects.

**Parameters:**

None

**Returns:**

Nothing

## 6.2.2 SMS.SendMessage method

*Function SendMessage(sDest As String, sMessage As String, [iSmsClass As fbSmsClass = fbSmsClassNormal]) As Boolean*

This method will send an SMS message to sDest containing text sMessage. The set of SMS settings to be used can be set by altering the SMSSettingSet property of the Mobile FBUS control. The iSmsClass parameter can be used to control the type of SMS message being sent.

### Parameters:

sDest -- Mandatory -- String containing destination cell phone number

sMessage -- Mandatory -- String containing the message to be sent

iSmsClass -- Optional -- SMS class (defaults to Normal). This parameter may have one of the following values:

fbSmsClassFlash      Message is shown as flash message (immediately show in main display of the telephone without user confirmation)

fbSmsClassNormal    Message is handled as normal SMS message.

fbSmsClass2          Same effect as fbSmsClassNormal in Nokia telephones.

fbSmsClass3          Same effect as fbSmsClassNormal in Nokia telephones.

### Returns:

True if the function succeeds

## 6.3 Sub-objects

### 6.3.1 SMS.DeliveryNotifications/Inbox/Outbox objects

These objects encapsulate the three main SMS message boxes, each having this property:

Property	Description
Count	Number of messages in box (read-only Integer). Messages have indexes between 1 and Count.

### 6.3.2 FBSmsMessage object

Each box contains a collection of message objects of type FBSmsMessage, each message having the following properties and method:

Property	Description
DateTime	Timestamp of message (read-only Date).
Destination	Destination of message (read-only String). In most phone types this property is only available in the OutBox.
LastError	Last FBUS error, 0 if last action was successful (read-only Integer).
Sender	Sender of message (read-only String). In most phone types this property is only available in the InBox.
SentRead	SentRead property returns true if the message is sent or read (read-only Boolean).
Text	Returns the message text (read-only String).
Method	Description
Delete	Deletes the message from the phone memory. You have to call SMS.Refresh to update the state of the SMS object after deleting messages.

For example, SMS.InBox(1).Text returns the message text of the first message in the InBox.

## 7 Logo object

This object can be used to manipulate the operator logo of the connected phone. The logo object has the following properties and methods:

Property	Description
Height	Returns the height of the current logo.
LastError	Returns the last error code.
LogoType	Returns/sets logo type (currently only operator logo).
NetCode	Returns/sets network code for the current operator logo.
Pixel	Returns/sets a pixel value in the current logo bitmap.
Width	Returns the width of the current logo.

Method	Description
Download	Downloads the operator logo from the phone.
ReadFile	Reads a logo from file.
SendAsSMS	Sends the current logo in an SMS message
Upload	Uploads the current logo to the phone.
WriteFile	Writes the current logo to file.

### 7.1 Properties

#### 7.1.1 Logo.Height property

*Integer - Read only*

This property returns the height of the current logo. 0 if no logo is loaded in the Mobile FBUS control. Mobile FBUS version 1.5 only supports operator logos that have a height of 14 pixels.

#### 7.1.2 Logo.LastError property

*Integer - Read only*

This property returns the last internal error that occurred when calling a Logo property or method. LastError is 0 when the last operation completed normally.

#### 7.1.3 Logo.LogoType property

*fbLogoType - Read/Write*

This property returns or sets the logo type for the current logo. Mobile FBUS version 1.5 only supports operator logos (type fbOperatorLogo); setting another type will result in an error being raised.

fbLogoType value	Meaning
FbNoLogo	No current logo present or load failure.
FbStartupLogo	Current logo is a startup logo (not supported in v1.5).
FbOperatorLogo	Current logo is an operator logo.
FbCallerLogo	Current logo is a caller logo (not supported in v1.5).

#### 7.1.4 Logo.NetCode property

*String - Read only*

This property returns or sets the network code for the current operator logo. If this code does not match the provider code of the network that the connected phone is currently using, the operator logo will not be visible when you upload it to the phone.

#### 7.1.5 Logo.Pixel property

*Pixel(x As Integer, y As Integer) As Integer - Read/Write*

This property returns or sets the value of one pixel in the operator logo. Currently only monochrome logos with pixel values of 0 or 1 are supported. The upper left corner of the logo has pixel coordinates (0, 0).

Parameters:

x -- Mandatory -- Horizontal coordinate, valid range between 0 and Logo.Width - 1

y -- Mandatory -- Vertical coordinate, valid range between 0 and Logo.Height - 1

### 7.1.6 Logo.Width property

*Integer - Read only*

This property returns the width of the current logo. 0 if no logo is loaded in the Mobile FBUS control. Mobile FBUS version 1.5 only supports operator logos that have a width of 72 pixels.

## 7.2 Methods

### 7.2.1 Logo.Download method

*Sub Download(iType As fbLogoType)*

This method downloads a logo from the phone to the Mobile FBUS control.

**Parameters:**

iType -- Mandatory -- Logo type to be downloaded:

FbStartupLogo	Current logo is a startup logo (not supported in v1.5).
FbOperatorLogo	Current logo is an operator logo.
FbCallerLogo	Current logo is a caller logo (not supported in v1.5).

The current version of Mobile FBUS only supports operator logos, so the value of iType must be fbOperatorLogo; passing another type will result in an error being raised.

**Returns:**

Nothing

### 7.2.2 Logo.ReadFile method

*Sub ReadFile(sFileName As String)*

This method reads a logo from file.

Supported file formats are NOL (Nokia Operator Logo) and NLM (Nokia Logo Manager file). Note that you could use a VB Image or PictureBox control and the [SetLogoPicture](#) method of the Mobile FBUS control to translate other bitmap file types to operator logos.

**Parameters:**

sFileName -- Mandatory -- Path/filename of file to be read

**Returns:**

Nothing

### 7.2.3 Logo.SendAsSMS method

*Sub SendAsSMS(ByVal sDestination As String)*

This method sends the current logo as an SMS message.

**Parameters:**

sDestination -- Mandatory -- Destination cell phone number

**Returns:**

Nothing

### 7.2.4 Logo.Upload method

*Sub Upload()*

This method uploads the current logo to the connected phone. Note that an operator logo will only be visible on the display of the phone if the [NetCode](#) property of the logo matches the [provider code](#) of the GSM network being used.

**Parameters:**

None

**Returns:**

Nothing

### 7.2.5 Logo.WriteFile method

*Sub WriteFile(ByVal sFileName As String)*

This method writes the current logo to file. The current version of Mobile FBUS will always use the NOL (Nokia Operator Logo) file format.

**Parameters:**

sFileName -- Mandatory -- Path/filename of file to be written

**Returns:**

Nothing